Real-Time Robot Vision on Low-Performance Computing Hardware

Lan, Gongjin; Benito-Picazo, Jesus; Roijers, Diederik M.; Dominguez, Enrique; Eiben, A. E.

published in
15th International Conference on Control, Automation, Robotics and Vision (ICARCV 2018)
2018

DOI (link to publisher)
10.1109/ICARCV.2018.8581288

document version
Publisher's PDF, also known as Version of record

Real-time Robot Vision on Low-performance Computing Hardware

Gongjin Lan, Jesús Benito-Picazo, Diederik M. Roijers, Enrique Domínguez, A.E. Eiben

Abstract— **Small robots have numerous interesting applications in domains like industry, education, scientific research, and services. For most applications vision is important, however, the limitations of the computing hardware make this a challenging task. In this paper, we address the problem of real-time object recognition and propose the Fast Regions of Interest Search (FROIS) algorithm to quickly find the ROIs of the objects in small robots with low-performance hardware. Subsequently, we use two methods to analyze the ROIs. First, we develop a Convolutional Neural Network on a desktop and deploy it onto the low-performance hardware for object recognition. Second, we adopt the Histogram of Oriented Gradients descriptor and linear Support Vector Machines classifier and optimize the HOG component for faster speed. The experimental results show that the methods work well on our small robots with Raspberry Pi 3 embedded 1.2 GHz ARM CPUs to recognize the objects. Furthermore, we obtain valuable insights about the trade-offs between speed and accuracy.**

I. Introduction

Recognizing other objects is a prerequisite for most robotic tasks. For small robots however, there are some specific constraints that make object recognition particularly challenging, i.e., limited computational resources, a small memory capacity, and limited physical size. For example, the modular robots proposed in [1], only have space inside to fit a small battery and a small computer with low computational power. The same is true for other types of robots like drones [2] and swarm robots [3]. These constraints make real-time robot vision on such small robots a difficult challenge. Of course, it can be argued that for any given volume, e.g., 50 cm^3, the computing power fitting in that volume will increase over time by the development of technology. However, the need for even smaller robots will likely also increase. For instance, robots that can navigate the human digestive system to recognize abnormalities such as wounds or cancer would be of great relevance. Thus, we foresee that the trend of squeezing more computing power in the same volume will go in parallel with the trend of willing to have smaller and smaller robots. This means that the challenge of real-time robot vision on low-performance computing hardware is likely to stay relevant in the foreseeable future.

In this paper we address this challenge in the context of our ongoing research on modular robots whose shapes are variable by evolution [4], [5], [6], but the 'head' that contains the battery and the onboard computer is fixed and limited in size [1]. After considering several alternatives, we have chosen the Raspberry Pi 3 for the head for it offers a good trade-off between size, computing power, power dissipation, and price. In addition, the Raspberry Pi is a general CPU and Linux platform with good compatibility that facilitates the portability of the algorithms to other hardware.

The visual task our robots need to handle is to recognize other robots in real-time in a standalone mode, without communicating with each other or a central mainframe. To solve this problem we present two methods, one based on Convolutional Neural Networks (CNN) and one based on the Histogram of Oriented Gradients (HOG) descriptor and a linear Support Vector Machine (SVM) classifier. To increase the efficiency we propose the Fast ROI Search (FROIS) algorithm to quickly find the Regions of Interest (ROI).

To assess how the methods work, we investigate the trade-offs between accuracy and speed, under the following conditions:

1) The speed on the Raspberry Pi 3 must be above 1.5 fps to meet the requirement that it is real-time.

2) The accuracy must be higher than 95%.

The remainder of this paper is structured as follows. In section II, we describe related work in object recognition, particularly in real-time object recognition for robots. A detailed description of our methods are described in section III. We present the experimental results, analyze and discuss these results in section IV. Last, we conclude this paper and provide an outlook on future research.

II. Related work

As one of the early real-time object recognition techniques P. Viola and M. Jones [7], [8] proposed the famous method of simple features and cascade AdaBoost classifier for rapid object detection. Although it worked fast on a desktop computer with Intel Pentium III, it operated on 384 by 288 pixels images and is still not fast enough for the Raspberry Pi. In 2005, N. Dalal and B. Triggs [9] proposed the classic solution of Histograms of Oriented Gradients (HOG) features and a Support Vector Machine (SVM) classifier for human detection. This method became popular in object recognition for not only human detection but also for applications in other fields. Although HOG features and linear SVM classifiers are fast and accurate, the exhaustive search component is computationally expensive so that it is not fast enough on low-performance computational hardware.

In recent years, CNNs became the state-of-the-art methods for object recognition [10], R-CNN[11], fast R-CNN[12], Faster R-CNN[13], YOLO[14], SSD[15]. Although these methods have high accuracy, they usually work on a GPU

Gongjin Lan, Diederik M. Roijers, A.E. Eiben are with the Department of Computer Science, VU University Amsterdam, The Netherlands. {g.lan, d.m.roijers, a.e.eiben}@vu.nl

Jesús Benito-Picazo, Enrique Domínguez are with the Department of Computer Languages and Computer Science, University of Málaga, Spain {jpicazo, enriqued}@lcc.uma.es

system, and are far from working on low-performance computational hardware. An interesting convolutional neural network for object recognition was proposed in [16]. Although this work optimized fast R-CNN on an embedded platform for real-time object recognition, it works at 1.85fps speed on the CPU and GPU system. Similarly, a low-complexity fully-convolutional neural network that works on a GPU platform was proposed in [17] for object recognition based on YOLO. Yet, none of [17] and [16] is fast enough on a Raspberry Pi 3.

Both early and recent popular methods have been widely and successfully applied in many areas, such as pedestrian recognition, face recognition, etc. However, there is a common issue, all of them require expensive computation, hence they only work on powerful computational hardware. A real-time solution on the DSP-based embedded system was proposed in [18]. This solution works only on the dedicated DSP hardware platform, not on a general CPU system. An exciting solution was proposed based on HOG features and SVM classifier for pedestrian detection at 135 fps on a desktop computer equipped with an Intel Core i7 870 and a GPU [19]. Similarly, [20] presented an implementation of vehicle recognition at 4 fps at a resolution of 1224 by 370 pixels based on HOG feature and linear SVM classifier. However, none of these methods is fast enough on low-performance systems like the Raspberry Pi 3.

Although some of the above methods have fast speed and high accuracy, they still do not work well given the constraints related to small robots. J. Wu et. al. proposed a real-time solution for human recognition based on CENTRIST feature [21] and linear SVM classifier, which achieves object recognition at 20fps on a mobile robot (PackBot) embedded a CPU of Intel 1.2GHz Core 2 Duo in [22]. Yet, this is still not fast enough for our hardware because the Intel 1.2GHz Core 2 Duo significantly outperforms a Raspberry Pi 3. In spite of this, it shows the state-of-the-art speed for real-time object recognition in robot vision.

In addition, there are many studies that work on different robots and hardware for real-time object recognition. An end-to-end deep vision network model was proposed to predict possible good grasps, which works in real-time on a Baxter robot at a rate of 80 frames per second using a GPU system [23]. [24] proposed an approach for robot detection and localization based on Convolutional Neural Networks (CNNs) for RoboCup soccer robots at 8 fps on a weak GPU system. [3] presented an experimental framework for exploiting vision in e-puck swarm robot by recognizing the Quick Response (QR) code. Although it has the same constraints as our system, it only works for recognizing QR codes. [2] proposed an automatic detection and tracking method for the drone based on deep neural networks at 1.6 fps on a GPU system. [25] used the exclusive Qualcomm Snapdragon Flight board embedded a 2.4GHz processor to implement a visual-inertial drone system for real-time moving object detection. Although this computational board has good performance and small size, it is a proprietary system exclusive for the drone and the method only recognizes moving objects.

An alternative approach was proposed in [26] that implemented real-time object recognition using wireless communication between the mobile robot and the servers. [27] implemented the near real-time object recognition for drones by offloading the computation onto an off-board computation cloud. Although using a server with powerful computational resources can be a feasible solution, several applications need methods that can operate independently without a server. For instance, the communication time between the robots and servers are affected by variations in wireless bandwidths and this can form a severe bottleneck [26].

In summary, existing methods usually rely on powerful computing resources. In spite of this, some of the algorithms using convolutional neural networks and linear SVM classifier are reasonably fast, yet not fast enough to implement real-time object recognition on a Raspberry Pi 3.

III. Methodology

The main requirements for our methods are that they work on the Raspberry Pi 3 in real-time, and with high accuracy. For that purpose, we propose the Fast ROI Search (FROIS) algorithm to quickly identify regions of interest (ROIs) for object recognition. The ROIs proposed by FROIS are then fed to a CNN or HOG and SVM classifier. We propose improved HOG features for added performance benefits. The work-flow diagram of our methods is shown in figure 1.

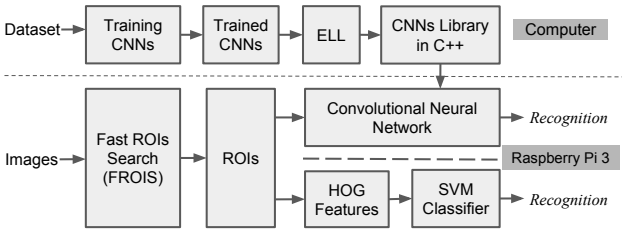
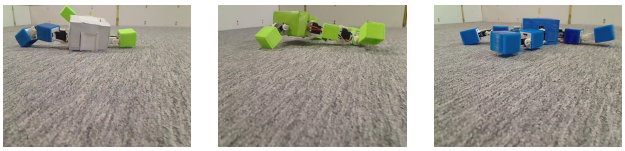


Fig. 1: The work-flow of our methods. The top shows the process that CNNs are trained and compiled into C/C++ library on the computer. The bottom shows the work-flow that our methods work on Raspberry Pi including 1) FROIS and CNNs, 2) FROIS, HOG features and SVM classifiers.

We test our approach on recognizing three different modular robots [1] shown in figure 2, called 'Baby', 'Gecko', 'Spider'. Real-time object recognition is essential for these robots to effectively perform tasks such as moving towards a destination [28] while avoiding collisions, or following one of the other robots. As can be seen from the figure, recognizing other robots is quite a different task from, e.g., distinguishing between different human faces or classifying genera of plants from images of leaves. The main challenge is not to reach the highest possible accuracy given as much computational resources as possible; it is to reach a high accuracy in real time. We discuss the robot dataset, as well as the additional task of recognizing coffee cups, in more detail in section IV.



Baby Gecko Spider

Fig. 2: Images of three real moving modular robots with different morphologies. The 'Baby' has 8 components in blue, white, and green color. The 'Gecko' has 7 green components. The 'spider' has 9 blue components.

A. *Fast ROIs search (FROIS)*

The region proposal and exhaustive search of current methods generate a large number of ROIs that are fed to the feature extractor and classifier. This is a key reason that the current methods for object recognition like R-CNN[11], fast R-CNN[12], Faster R-CNN[13], take prohibitively much computation time for real-time object recognition. In addition, algorithms for region proposal like selective search [29] also take a lot of computation time. In robot vision, the objects usually have simple and fixed features, and in particular fixed color features. We therefore propose the Fast ROIs Search algorithm to quickly find the ROIs based on the color features of objects.

First, FROIS detects and segments the images into the valid regions with the different colors. The HSV based color recognition is best suited for the images with less complex background [30], in particular uniform background for the environment of small robots.[1] Therefore, we convert the original RGB images into HSV. Then, the colors of pixels in the images are detected by matching HSV threshold values, given in table I, including Red, Green, Blue, Yellow, Black. We optimized these HSV threshold values by testing the color samples of various objects in the real environment, i.e., the environment the robots operate in. Therefore, the color detection based on table I are more practical and adaptable in the real environment. Although we show the HSV threshold value of 5 common colors in table I, this table can be extended to more colors for different objects.

colors	H	S	V
Red	0∼10 170∼180	160∼255	120∼255
Green	35∼40 41∼59	140∼255 69∼255	104∼255
Blue	99∼121	120∼255	57∼211
Yellow	26∼32	130∼255	150∼255
Black	3∼180	40∼235	6∼48

TABLE I: The HSV threshold values of colors.

After the color detection, we need to segment the images into valid regions with different colors. The algorithm detects the contours of continuous regions with the same color. Then, FROIS calculates the areas of continuous regions according to the contours. Invalid regions that may be noise can be filtered by removing small area regions, using a minimal threshold value. The images thus are segmented into valid color regions. The number, size, and position of valid color regions are different for different objects. To make a clear description, we adopt the modular robot 'baby' as the object to show how the algorithm proposes the ROIs based on the valid color regions. The valid green and blue regions of the modular robot 'baby' after the segmentation are shown in figure 3.

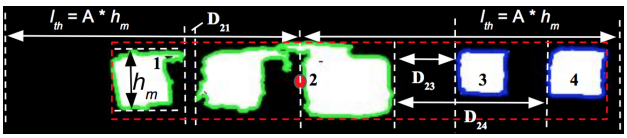


Fig. 3: Segmentation image of the modular robot 'Baby'. The regions 1 and 2 (green contours), region 3 and 4 (blue contours) are valid regions. l_{th} is a threshold value. A is a constant. The left and right white dashed lines are the edges of l_{th}. The red dashed rectangle is the ROI of a modular robot 'baby'.

Second, the algorithm detects ROIs based on the valid color regions. The algorithm searches the center regions from all of the regions. The minimum distance between each region is calculated. For instance, D_{23} in figure 3 is the minimum distance between region 2 and 3. The sum of the distances from a region to the nearest two regions, is denoted D_n. For instance, the sum of distances of regions 2 is $D_2 = D_{21} + D_{23}$. The method chooses the region with the minimum D_n as the center region. In figure 3, region 2 is the center region because D_2 is less than $D_i, i \neq 2$. The regions around a center region belong to the same ROI of the objects if the distances to the center region less than a threshold value. Therefore, determining a suitable threshold value for the distance from a region to a center region is important to find the right ROIs of the objects, denoted l_{th} in figure 3. However, it is difficult to find a fixed threshold value. The color regions from different objects have different distances to the center regions, and even the same regions of an object have different distances to the center regions when the objects rotate. In spite of this, we noted that the median height of regions are basically invariant or change only very slightly as the objects do not rotate vertically. Furthermore, the distance from other regions to the center regions is proportional to the median height of the regions when the objects display at different scales in the images. We denote the median height as h_m in figure 3. l_{th} is proportional to h_m whatever the scales of the objects are, because h_m is an adaptive value based on the images captured from different views and distances. Taking the modular robot 'Baby' as an example, we set $l_{th} = A * h_m$, A equals 5. For the modular robot 'Baby' in figure 3, the green and blue regions belong to

[1]We tested other color models like RGB, YCbCr, and YUV, but the HSV based color detection has the best performance in our experiments.

the same ROI because their minimum distance to the region 2 are less than l_{th}. That is, the ROI is the minimum rectangle region that covers the color regions of the same ROI. For instance, the red dashed rectangle is the ROI in figure 3. The full FROIS algorithm is provided in algorithm 1.

Last, although we used a modular robot 'baby' as the object to address the FROIS algorithm, it works well for the different objects that have basic fixed features in robot vision. The ROIs of the objects can be found quickly, which significantly reduces the computation time. We show that FROIS quickly finds the ROIs of the modular robots (and coffee cups) in section IV.

Algorithm 1: The Fast ROIs Search algorithm

Input: RGB image
Output: ROI[r] of the objects

while *image* **do**
  initialize A, D_{center}, k, r;
  Detect and segment the image with color, n regions;
  for $i \in n$ **do**
    for $j \in n$ **do**
      calculate distance $D_{ij}, i \neq j$;
    end
    ranking D_{ij} as D_{i1}, D_{i2}, ..., D_{in};
    $D_i = D_{i1} + D_{i2}$;
    if $D_i > D_{center}$ **then**
      center = i;
    end
  end
  calculate h_m;
  for $i \in n, i \neq center$ **do**
    if $D_{i\ center} < A * h_m$ **then**
      coor[k][$(min_x, min_y), (max_x, max_y)$];
      k = k +1, remove i from n;
    end
    ROI[r][$min(min_x, min_y)$,$max(max_x, max_y)$];
  end
  return ROI[r];
  r = r + 1;
  if $n >= 2$ **then**
    repeat 4 to 22;
  end
end

B. The Improved HOG features and SVM classifier

The ROIs of the objects are proposed by FROIS algorithm. However, ROIs need to be further analyzed because ROIs may or may not contain the right objects. Therefore, the ROIs are fed to the feature extractor and classifier. This work presents two methods for feature extraction and classification, i.e., CNNs [31] and the combination of HOG and SVM [9]. The method of HOG descriptor and SVM classifier, proposed by N. Dalal and B. Triggs, is not only popular in human detection, but is widely applied in the field of object recognition. The traditional HOG features and SVM classifier recognize the objects using the exhaustive search to scan the images for objects of different positions and sizes, which takes much computation time. In this subsection, we address how the HOG features and linear SVM classifiers work together with the FROIS algorithm for real-time object recognition on low computational resources. We propose an improvement for HOG features, making them even faster.

The different parameters of HOG features have different performance for accuracy and speed. We trained and tested SVM classifier with different parameters of the HOG features, include block size, block stride, cell size, and bins. The main parameter configuration is shown in table III. Although the HOG features and the limited ROIs are not computationally expensive, we note that the computing HOG features can be optimized by employing the techniques of the lookup table. In this way, the computation can be reduced since retrieving a value from memory is often faster than undergoing an "expensive" computation. For the HOG features, the magnitude and angle [9] for each pixel of an ROI needs to be calculated. The possible values for magnitude and angle are bounded. The magnitudes are in the interval $(-255, 255)$, and the angles in $(0°, 360°)$. Furthermore, the magnitude and angle of each pixel are used to vote into the histograms of oriented gradients in 9 bins. Therefore, we created a lookup table containing the possible values of magnitude and angle, which is sufficient to vote accurately. Both in theory and practice, this optimization reduces the computation time. Last, we used the standard linear SVM classifier from the OpenCV library to classify the objects based on the HOG features.

C. The Convolutional Neural Network

To investigate the available trade-offs that exist between speed and accuracy, we use different feature descriptors and classifiers to test speed and accuracy. We implement the CNNs on the Raspberry Pi 3 for real-time object recognition. The CNNs take the ROIs that are proposed by the FROIS algorithm as input. Although CNNs have the state-of-the-art performance for object recognition, it is well known that the CNNs are difficult to implement on the low computational hardware like the Raspberry Pi. However, Darknet [32] and the new Embedded Learning Library (ELL[2]) provides the possibility to implement CNNs on the Raspberry Pi.

Darknet is an open source neural network framework written in C, optimized for speed. ELL is an embedded AI and machine learning toolbox developed at Microsoft Research, which allows us to design and deploy CNNs onto resource constrained platforms and small single-board computers, like the Raspberry Pi, Arduino, and micro:bit. We trained CNNs with different topologies to identify the available trade-offs between speed and accuracy. The trained CNNs are compiled into a C/C++ library by ELL, subsequently deployed onto the Raspberry Pi 3. The work-flow is shown in figure 1.

[2] https://github.com/Microsoft/ELL

IV. Experiments

We aim to achieve real-time object recognition on low computation hardware. Therefore, we compare the performance of our proposed methods, i.e., FROIS in combination with either CNNs or HOG and SVM, in terms of accuracy and speed. All methods were implemented in C/C++. We implement and train CNNs using Darknet and the linear SVM classifier using OpenCV, and test them on the Raspberry Pi 3.

We test the algorithms in different situations with the modular robots, including those with occlusions. The dataset we use for this is outlined in section IV-A, and the results in section IV-B. Furthermore, to validate the generality and scalability of our methods for real-time object recognition in robot vision, we also test whether our methods can recognize the other objects. For this we use coffee cups in section IV-C.

A. The Robot Dataset

In object recognition, the comprehensiveness of the dataset is directly related to the performance. We took a large number of images for three modular robots from different views and distances to create a comprehensive dataset[3]. All of the images were taken by the modular robots while moving. We divided the samples into training and a testing dataset, as shown in table II. We use the same datasets to train and test CNNs and SVMs. All of the samples have a resolution of 112 by 32 pixels.

	Baby	Gecko	Spider
Training	3246	2943	3204
Testing	2211	2131	2193

TABLE II: The statistics of the dataset.

B. Robot Results

We train and test CNN and SVM classifiers for different parameter configurations (Table III). The accuracy and computation time are averaged over ten frames at different times. To determine the accuracy we ran the methods (FROIS+HOG+SVM or FROIS+CNN) on a testing dataset on the computer and to determine the speed we ran the methods on video input on the Raspberry Pi 3 and Raspberry Pi camera V2. The top three accuracies across parameter configurations for HOG and SVM, are over 98%. Those configurations of HOG and SVM that have higher accuracies also have higher computation times. The best HOG and SVM configuration (the gray row) achieves a real-time object recognition of 99.1% accuracy and around 10 frames per second on the Raspberry Pi 3. The CNN with 8 convolutional layers achieved 94.51% accuracy and 1.6 frames per second. Although the accuracy of CNNs can be improved by adding layers, this leads to lower speeds than required for operation in real-time. Furthermore, the accuracy and speed of the best CNN with 7 convolutional layers performs well at a 95.99% accuracy and 1.8 frames per second. We thus conclude that while CNNs are more suitable for high accuracy object recognition on more powerful hardware, for object recognition on low computing hardware, HOG and SVM is actually more promising. In particular the faster speed of HOG and SVM makes it a more suitable approach. We observe that accuracy and speed are two conflicting objectives, and that several possible trade-offs exist between accuracy and speed.

We tested the computation times, of the FROIS algorithm, HOG features, SVM classifier, and CNNs, individually as well in figure IV. The HOG with the techniques of lookup table takes less computation time than the original HOG. The SVM classifier is significantly faster than the CNNs. In addition, the FROIS is indeed fast at about 17ms.

Using the performance results of table III, we would choose the gray configuration of HOG and SVM as the best due to high (99.9%) accuracy and fast speed (10 fps). A typical result of recognizing the modular robots 'baby', 'gecko', 'spider' using this configuration is shown in figure 4. The three modular robots in the figure are recognized accurately. The 'spider' is recognized even though about a third of 'spider' is occluded by a black box. In figure 5, the region of modular robot 'baby' (middle) is proposed as the ROI (red rectangle) by FROIS algorithm, but it is recognized as 'Not Robot' because it is strongly occluded by the black box, and only two loose components are visible. It is therefore not clear whether this is a robot or just loose blocks. We therefore conclude that our methods work well, even in occlusion situations.

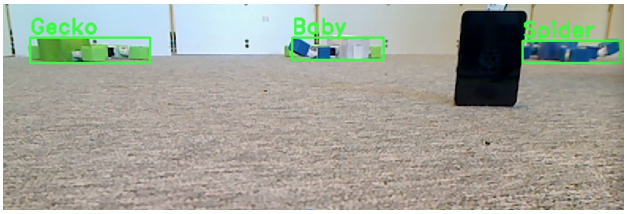


Fig. 4: The recognition of three modular robots. The left is 'Gecko'. The middle is 'Baby'. The right is 'Spider'. The green texts above bounding box are the names of recognized robots. The green bounding boxes are the regions of recognized robots. The black box occludes a piece of 'Spider'. please note that black color is not a color feature of the modular robots 'baby'.

We compared to our methods to standard HOG and SVM [9] on the Raspberry Pi 3 with Raspberry Pi camera v2 for recognizing modular robots 'baby'. The computation time of standard HOG and SVM was 4740.954 ms (average) per frame, which is prohibitively slow for real-time computation. In addition, our methods recognize the position of the modular robots more accurately. In figure 6, the modular robot 'baby' was recognized by HOG and SVM with exhaustive search rather than the FROIS algorithm. Note that the same robot is recognized twice in different positions. Furthermore, the right red bounding box does not accurately reflect the

[3] https://bitbucket.org/langong/robotsdataset/downloads/

The configurations				Accuracy(in percent)				Computation time(ms/f)	
blocksize	blockstride	cellsize	bins	baby	gecko	spider	average	original	improved
(8, 8)	(4, 4)	(4, 4)	9	0.996834	0.990615	0.974191	0.991099	99.3862	98.1998
(8, 8)	(4, 4)	(4, 4)	6	0.984622	0.977475	0.982216	0.983594	98.2688	97.1797
(8, 8)	(8, 8)	(4, 4)	9	0.982813	0.974191	0.98404	0.984659	95.9153	94.3975
(8, 8)	(8, 8)	(4, 4)	6	0.917232	0.933834	0.952576	0.947332	95.5965	94.0269
(16, 16)	(8, 8)	(8, 8)	9	0.928991	0.935711	0.943	0.947459	96.412	94.7658
(16, 16)	(16, 16)	(8, 8)	9	0.848033	0.870483	0.877793	0.886323	95.4248	93.9846
8 layers CNN				0.911111	0.924419	1.000000	0.945177	634.3959	
7 layers CNN				0.896296	0.994186	0.989071	0.959851	568.3451	
6 layers CNN				0.681481	0.988372	1.000000	0.889951	545.9009	

TABLE III: The experimental results w.r.t. accuracy and speed for different parameters of our methods. The top half are the accuracy and the speed for the HOG and SVM with different parameters. The computation time includes the full procedure of FROIS, HOG, and SVM. Note that it is not including the time of reading images.

	$t_{ROI}(ms)$	$t_{HOG}(ms)$	$t_{SVM}(ms)$
original	17.1086	8.101	0.6035
improved		5.9577	
	t_{ROI}	t_{CNN}	
CNN	17.1664	537.0053	

TABLE IV: Distribution of computation time of our methods. The CNN is the 8 layers CNN in table III. NB: it takes about 80ms to read in the image, resulting in the higher processing times of table III.

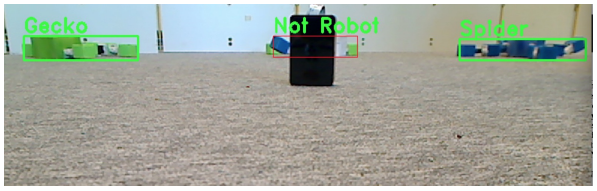


Fig. 5: The recognition for an occlusion situation. The center part of robot 'Baby' (middle) is occluded by a black box. The red bounding box is the ROI proposed by FROIS algorithm. The green text above the red bounding box is 'Not Robot'.

robot's position. This problem does not occur our methods because as the ROIs are proposed more effectively and accurately by the FROIS algorithm. We thus conclude that our methods are key for real-time object recognition with low computational resources.

C. Coffee Cups

To validate the generality and scalability of our methods, we also test recognizing common coffee cups that we took from the automatic coffee machine. In this trial, we detected black color for proposing the ROIs, and modified the threshold value l_th from $l_th = 5 * h_m$ to $l_th = 3.5 * h_m$, to reflect the proportions of the cups. Then, we created the dataset for the coffee cups and trained an SVM classifier for real-time

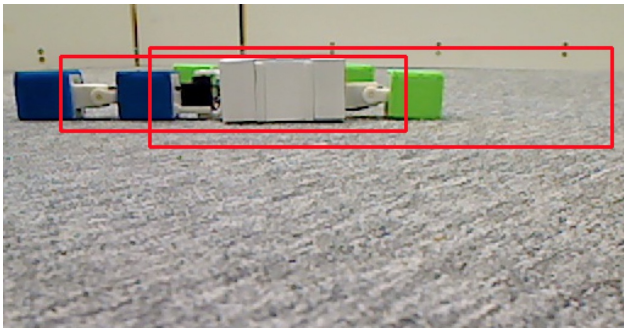


Fig. 6: The recognition of the HOG features and linear SVM classifier with the exhaustive search. The red bounding boxes are the regions recognized as a 'baby' robot.

coffee cup recognition on low-computation hardware. Figure 7 shows that our methods work well for the recognition of the cups as well. The green region is proposed as ROI by FROIS algorithm and recognized as a cup by HOG and SVM. Although the half of the right cup is proposed as an ROI (the red bounding box), half of the cup is occluded and thus recognized it as 'nothing' by HOG and SVM.

V. Conclusions

In this paper, we addressed the challenge of real-time robot vision on low-performance computing hardware. Our robots have a small 'head' containing a Raspberry Pi 3 and they need to recognize other objects in real-time in a standalone mode, without communicating with each other or a central mainframe using a Raspberry Pi camera V2. To solve this problem we proposed the FROIS algorithm to quickly search the ROIs of the objects. This algorithm was tested in combination with a CNN and a method based on HOG and SVM. The tests included several algorithm variants and the winning combination (FROIS+HOG+SVM) achieved a high accuracy (99.1%) and fast speed (10 fps). The recognition of other objects validates the generality and

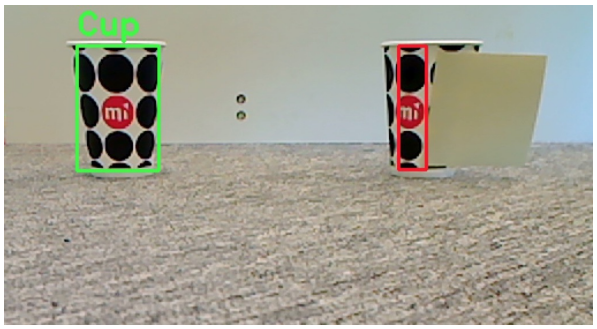


Fig. 7: The recognition of the common coffee cups. The green bounding box is the recognized regions by HOG and SVM. The red bounding box on the right half-cups is the region of interests that proposed by color feature, but not recognized as the cups.

scalability of our methods for real-time object recognition in robot vision. We thus conclude that this method can be applied to real-time object recognition in robots with low-performance computing hardware. Ongoing and future work concerns more and more demanding test cases, including more complex environments. This will provide more significant support for the applicability of small robots for interesting tasks that require vision capabilities.

REFERENCES

[1] M. Jelisavcic, M. de Carlo, E. Hupkes, P. Eustratiadis, J. Orlowski, E. Haasdijk, J. E. Auerbach, and A. E. Eiben, "Real-world evolution of robot morphologies: A proof of concept," *Artificial Life*, vol. 23, no. 2, pp. 206–235, May 2017.

[2] S. Han, W. Shen, and Z. Liu, "Deep Drone : Object Detection and Tracking for Smart Drones on Embedded System."

[3] S. Alers, B. Ranjbar-sahraei, S. May, K. Tuyls, and G. Weiss, "An experimental framework for exploiting vision in swarm robotics," in *In ADAPTIVE 2013, The Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2013, pp. 83–88.

[4] A. Eiben and J. Smith, "From evolutionary computation to the evolution of things," *Nature*, vol. 521, no. 7553, pp. 476–482, May 2015.

[5] A. E. Eiben, "Evosphere: The world of robot evolution," in *Theory and Practice of Natural Computing*, A.-H. Dediu, L. Magdalena, and C. Martín-Vide, Eds. Cham: Springer International Publishing, 2015, pp. 3–19.

[6] A. Eiben, N. Bredeche, M. Hoogendoorn, J. Stradner, J. Timmis, A. Tyrrell, and A. Winfield, "The triangle of life: Evolving robots in real-time and real-space," in *Advances In Artificial Life, ECAL 2013*, P. Liò, O. Miglino, G. Nicosia, S. Nolfi, and M. Pavone, Eds. MIT Press, 2013, pp. 1056–1063.

[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–511–I–518 vol.1.

[8] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 580–587.

[12] R. Girshick, "Fast r-cnn," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 1440–1448.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

[16] H. Mao, S. Yao, T. Tang, B. Li, J. Yao, and Y. Wang, "Towards real-time object detection on embedded systems," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[17] S. Tripathi, G. Dane, B. Kang, V. Bhaskaran, and T. Nguyen, "Lcdet: Low-complexity fully-convolutional neural networks for object detection in embedded systems," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, July 2017, pp. 411–420.

[18] C. Arth and H. Bischof, "Real-time object recognition using local features on a dsp-based embedded system," *Journal of Real-Time Image Processing.*, vol. 3, pp. 233–253, 2008.

[19] L. Van Gool, "Pedestrian detection at 100 frames per second," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ser. CVPR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 2903–2910.

[20] S. Nawaz, "Hog-svm car detection on an embedded gpu," Master's thesis, 11 2015.

[21] J. Wu and J. M. Rehg, "Centrist: A visual descriptor for scene categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1489–1501, Aug 2011.

[22] J. Wu, C. Geyer, and J. M. Rehg, "Real-time human detection using contour cues," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 860–867.

[23] D. Guo, F. Sun, T. Kong, and H. Liu, "Deep vision networks for real-time robotic grasp detection," *International Journal of Advanced Robotic Systems*, vol. 14, no. 1, p. 1729881416682706, 2017.

[24] S. Luo, H. Lu, J. Xiao, Q. Yu, and Z. Zheng, "Robot detection and localization based on deep learning," in *2017 Chinese Automation Congress (CAC)*, Oct 2017, pp. 7091–7095.

[25] C. Huang, P. Chen, X. Yang, and K. Cheng, "REDBEE: A visual-inertial drone system for real-time moving object detection," *CoRR*, vol. abs/1712.09162, 2017.

[26] Y. Nimmagadda, K. Kumar, Y. H. Lu, and C. S. G. Lee, "Real-time moving object recognition and tracking using computation offloading," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 2449–2455.

[27] J. Lee, J. Wang, D. Crandall, S. Šabanović, and G. Fox, "Real-time, cloud-based object detection for unmanned aerial vehicles," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, April 2017, pp. 36–43.

[28] G. Lan, M. Jelisavcic, D. M. Roijers, E. Haasdijk, and A. E. Eiben, "Directed locomotion for modular robots with evolvable morphologies," in *Parallel Problem Solving from Nature – PPSN XV*, A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, Eds. Cham: Springer International Publishing, 2018, pp. 476–487.

[29] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[30] K. B. Shaik, P. Ganesan, V. Kalist, B. Sathish, and J. M. M. Jenitha, "Comparative study of skin color detection and segmentation in hsv and ycbcr color space," *Procedia Computer Science*, vol. 57, pp. 41 – 48, 2015, 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015).

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[32] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.